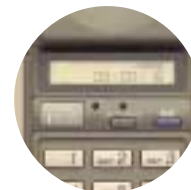


System-Level Design and Rapid Prototyping for TI DSPs



Ken Karnofsky
The MathWorks, Inc.
DSPSFest 2000
August 4, 2000

MATLAB
SIMULINK

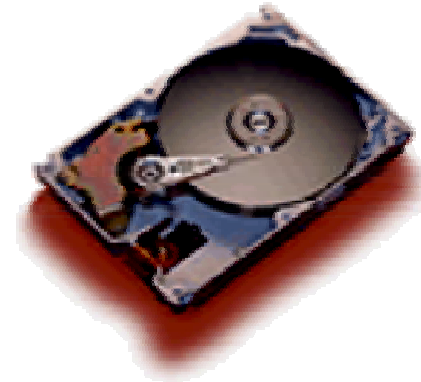
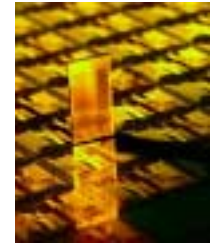


AGENDA

- **DSP Design Process Challenges**
- **MathWorks System-Level Design Solutions**
- **Texas Instruments DSP Developer's Kit**

Design Challenges

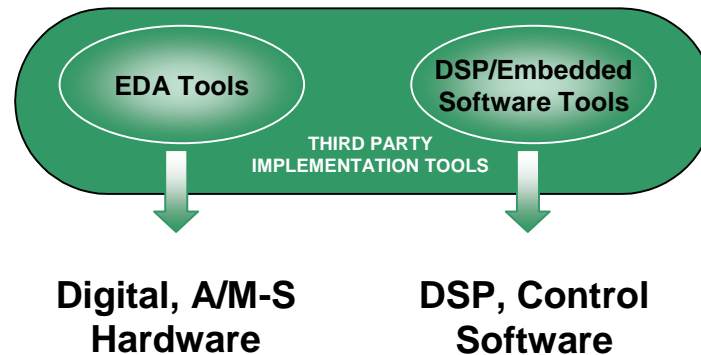
- **Time-to-market pressure**
 - Shorter product cycles
 - Increasing complexity
 - Shortage of DSP expertise
- **Traditional Process Limitations**
 - Gap between algorithm research and implementation
 - Poor integration of component design teams
 - System verification/testing occurs too late



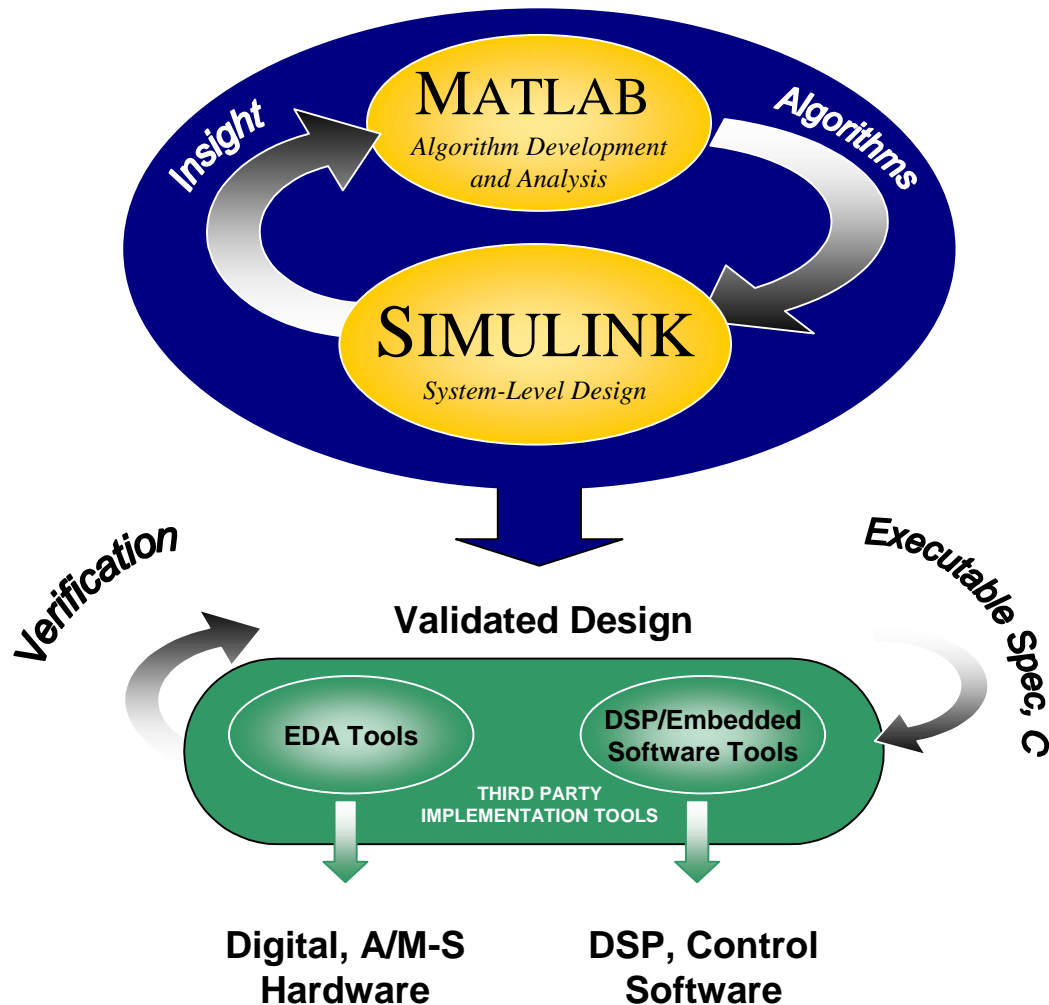
Design Flow Problem

**MATLAB, C,
other**

- **Ambiguous specifications**
- **Error-prone manual re-coding**
- **Design flaws detected too late**
- **Design failure risk too high**



Simulink System-Level Solution



Design Team Integration Problem

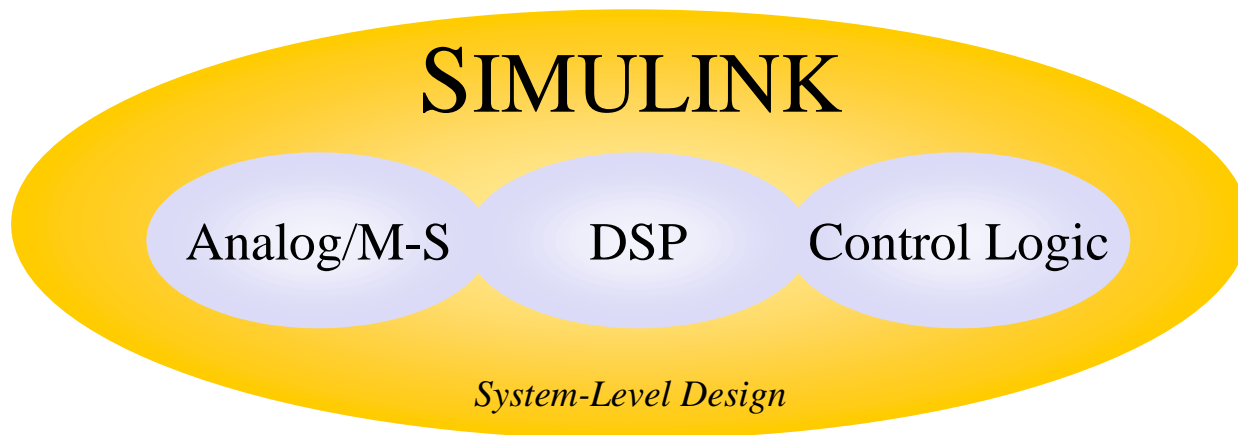
Analog/M-S

DSP

Control Logic

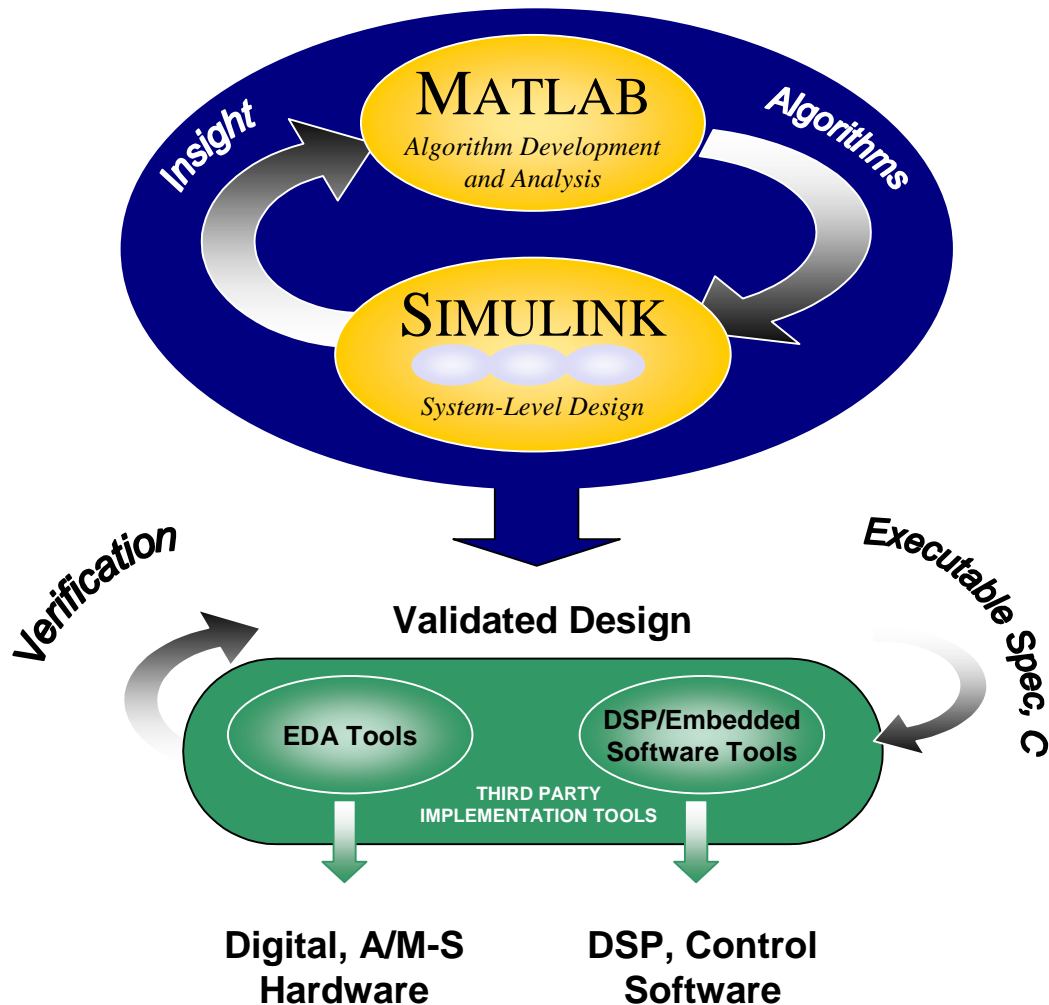
- **Independent design teams and tools**
- **Can't simulate component interactions**
- **Can't test whole system**
- **Expensive over-design**

Simulink Integrated Design Solution



- **Common tool for entire design team**
- **Simulate behavior of whole system**
- **Model component interactions**

Simulink Design Environment



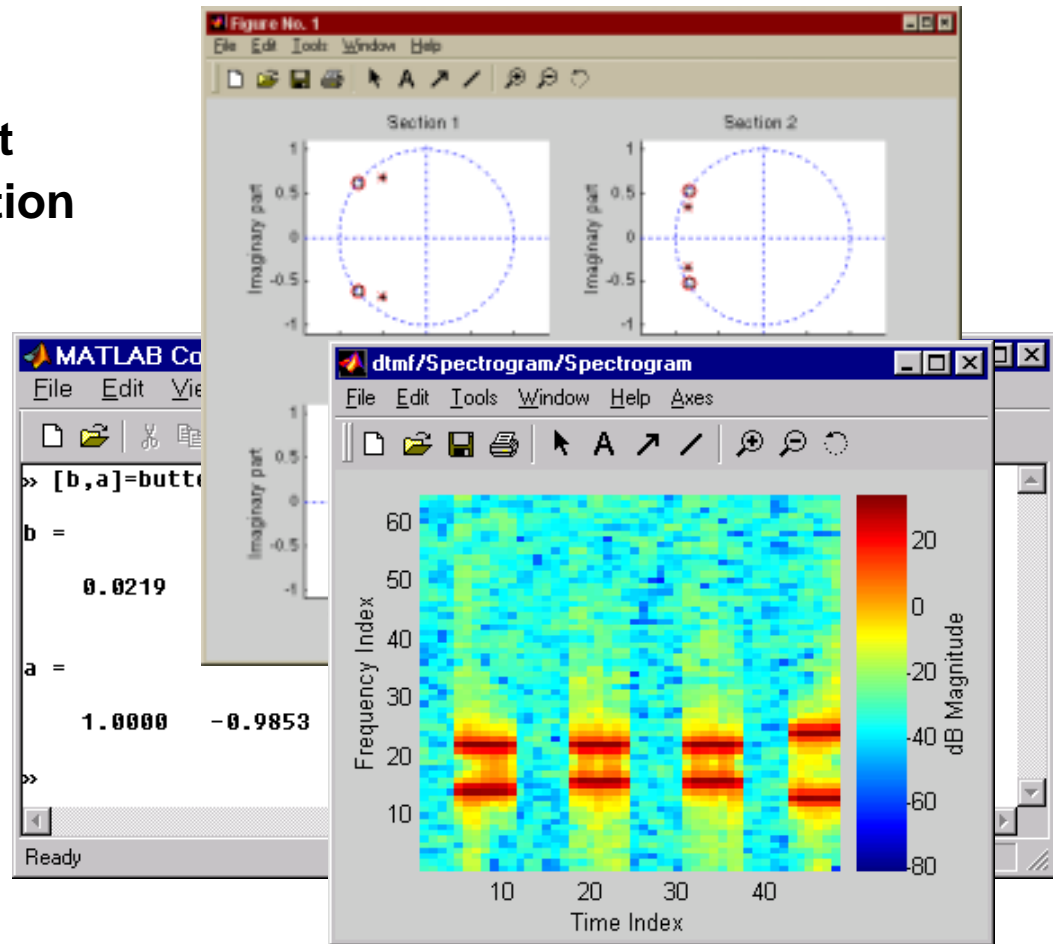
- Test entire system
- Model component interactions
- Reduce design risk
- Reduce time-to-market

The MathWorks System-Level Design Environment

MATLAB - The Language of DSP Algorithms

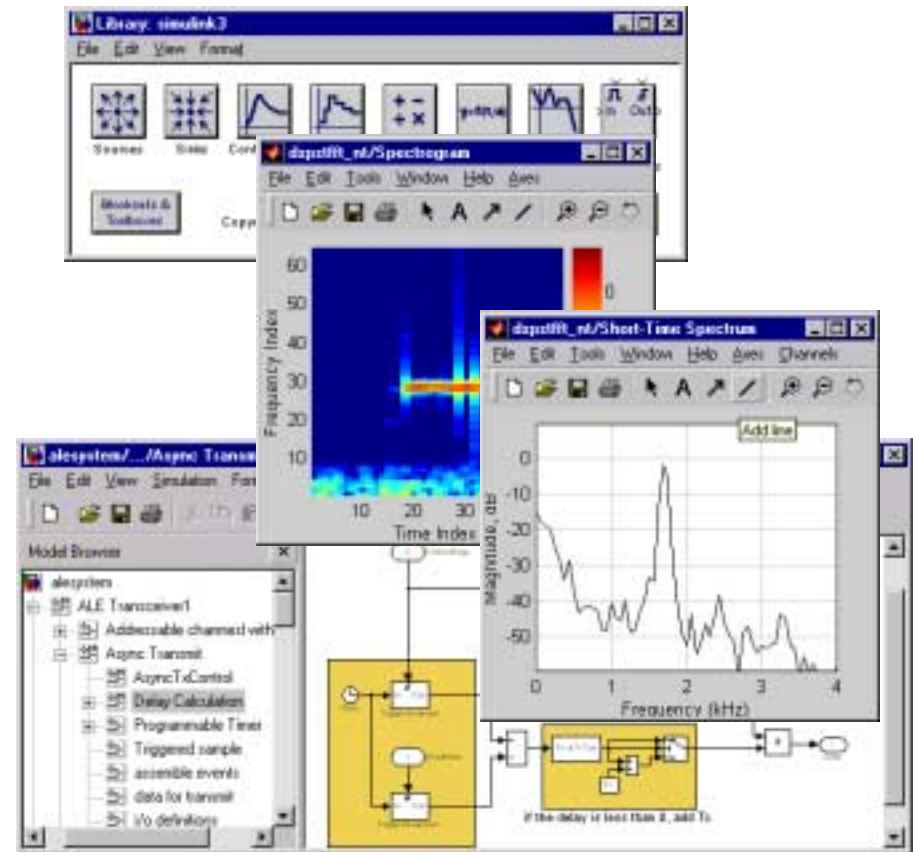
- **MATLAB**
 - Algorithm development
 - Analysis and visualization
 - Open and extensible

- **MATLAB Toolboxes**
 - Signal Processing
 - Quantized Filtering
 - Image Processing
 - Wavelet
 - Control Systems
 - Data Acquisition
 - and many more



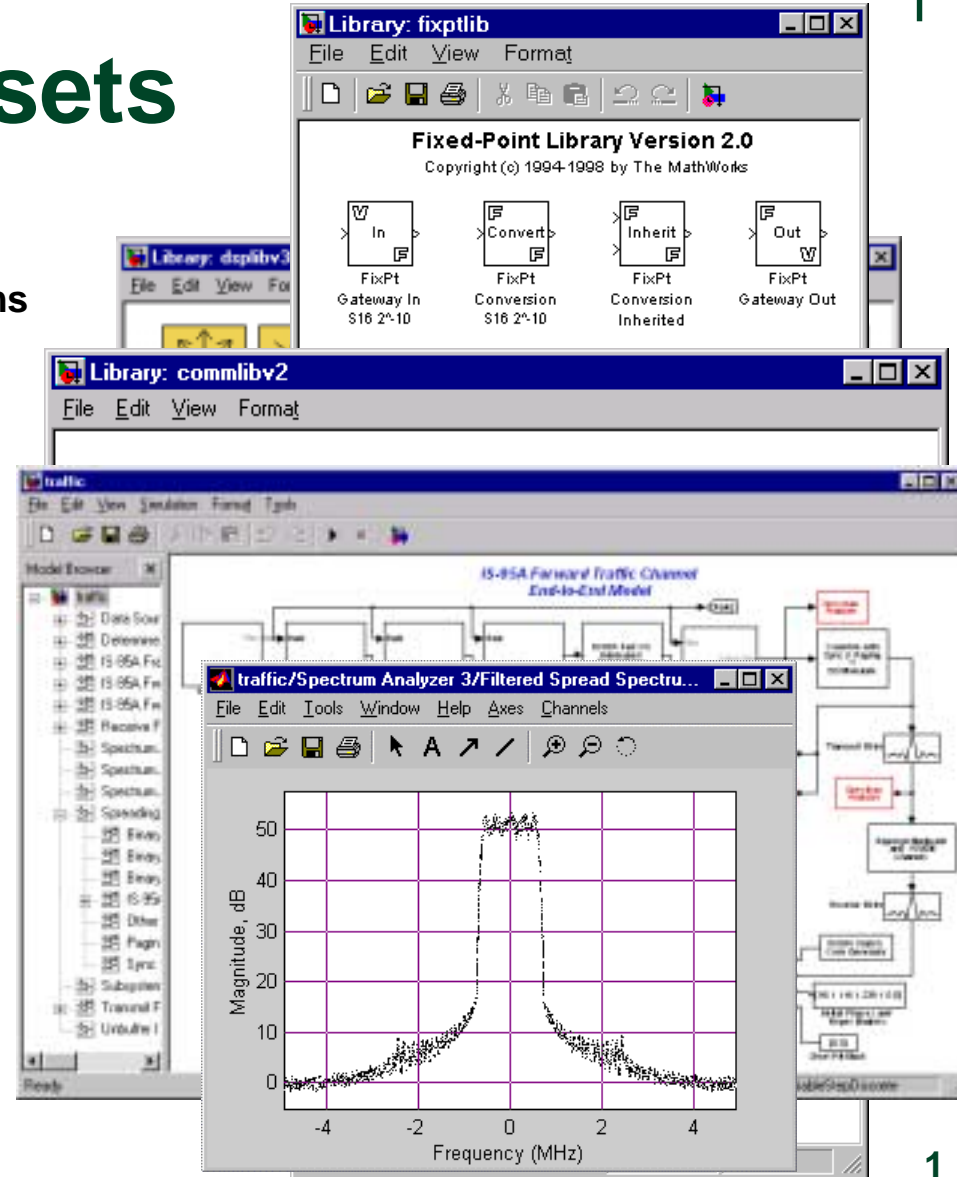
Simulink

- Hierarchical block diagrams
- Dynamic simulation
- Digital, analog/mixed signal
- Visualize signals
- Co-develop with C code
- Rapid prototyping
- Integrated with MATLAB



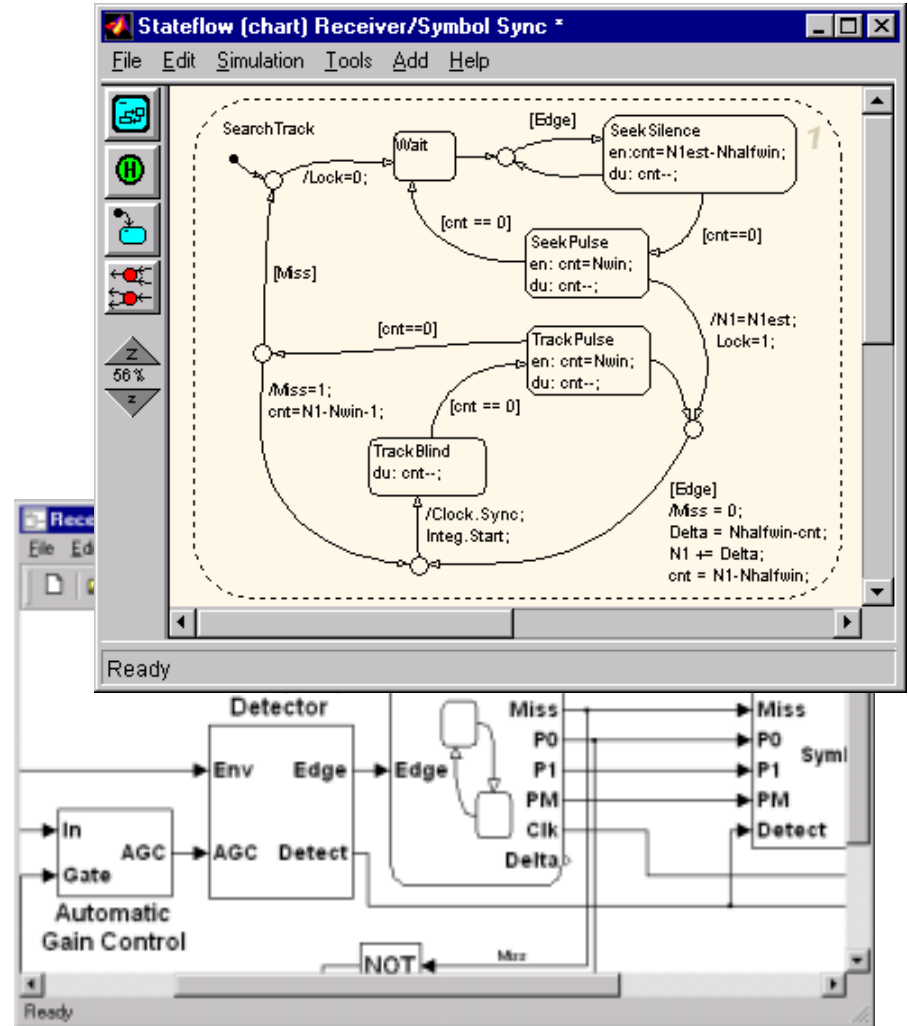
Simulink Blocksets

- **DSP**
 - Filtering, math, estimation, transforms
 - Multi-rate
 - Multi-channel, frame-based
- **Fixed Point**
 - Bit-true 1-128 bits
 - Scaling, rounding and overflow
 - Include own bit-true code
 - Filter wizard
- **Communications**
 - Modulation, coding, synchronization
 - Channel models
 - BER/BER system testing
- **CDMA Reference Blockset**
 - Validated for IS-95A standard
 - Springboard for 3G designs



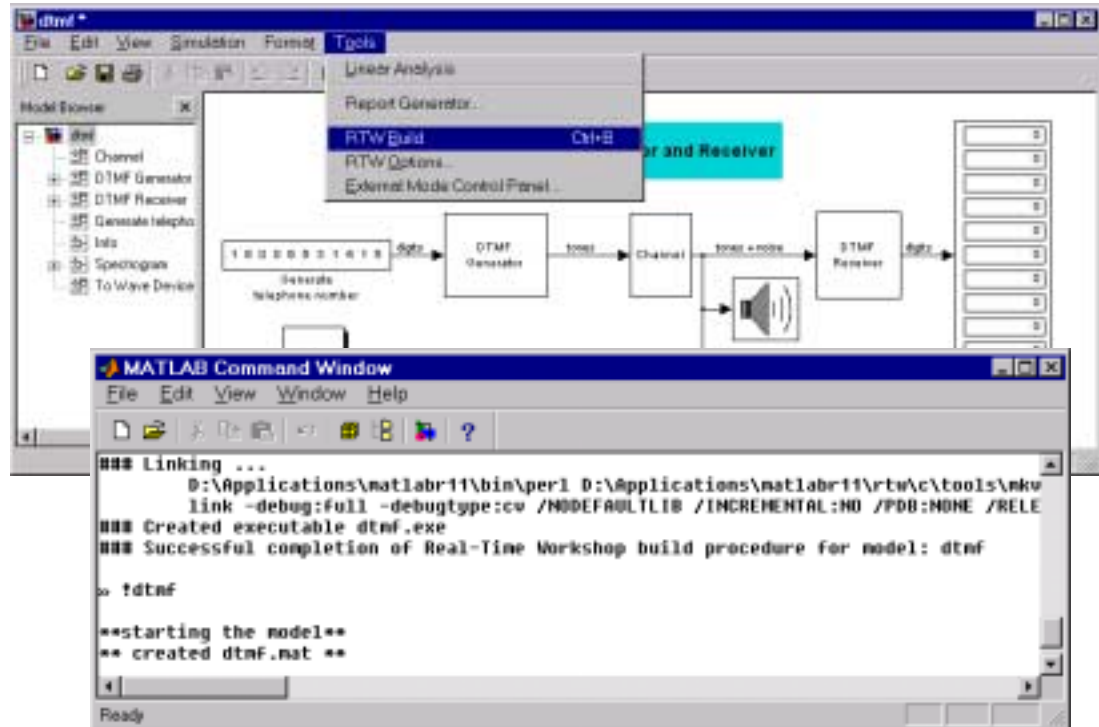
Stateflow

- Integrated with Simulink
- Finite state machines
- Flow diagrams
- Event driven systems
 - Control logic
 - Protocols
 - Synchronization
- Stateflow coder
 - Integer C code generation

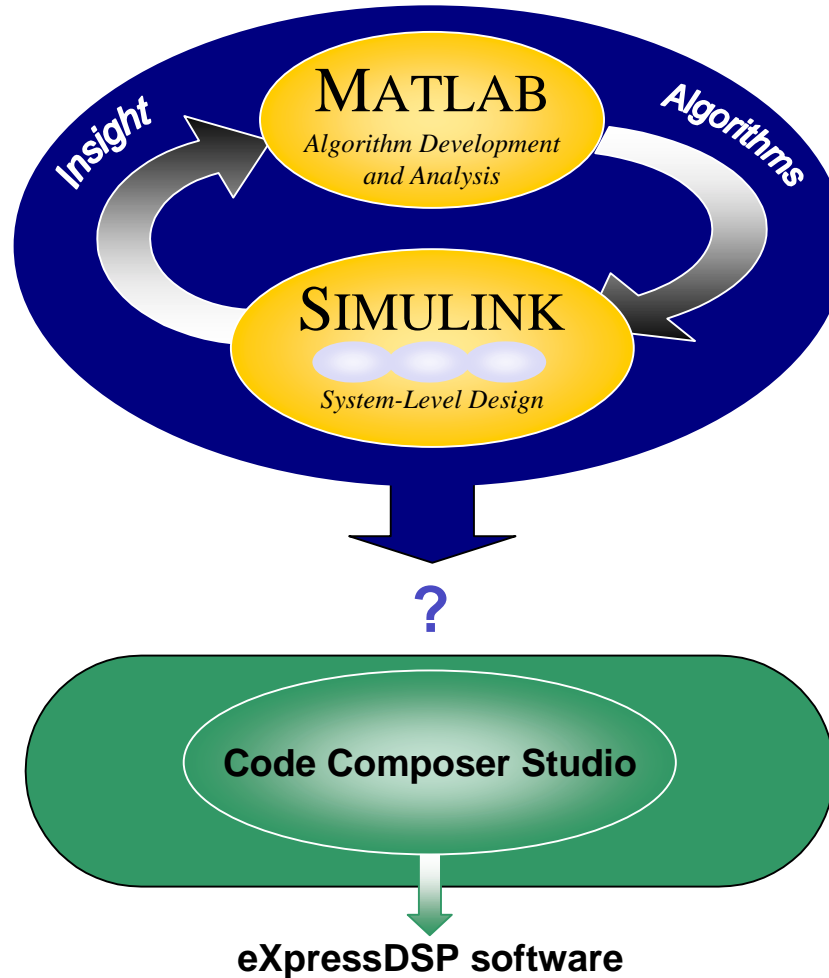


Real-Time Workshop

- Code generation from Simulink
 - ANSI C
 - Target-specific customization
- Real-time rapid prototyping
- High performance standalone simulations

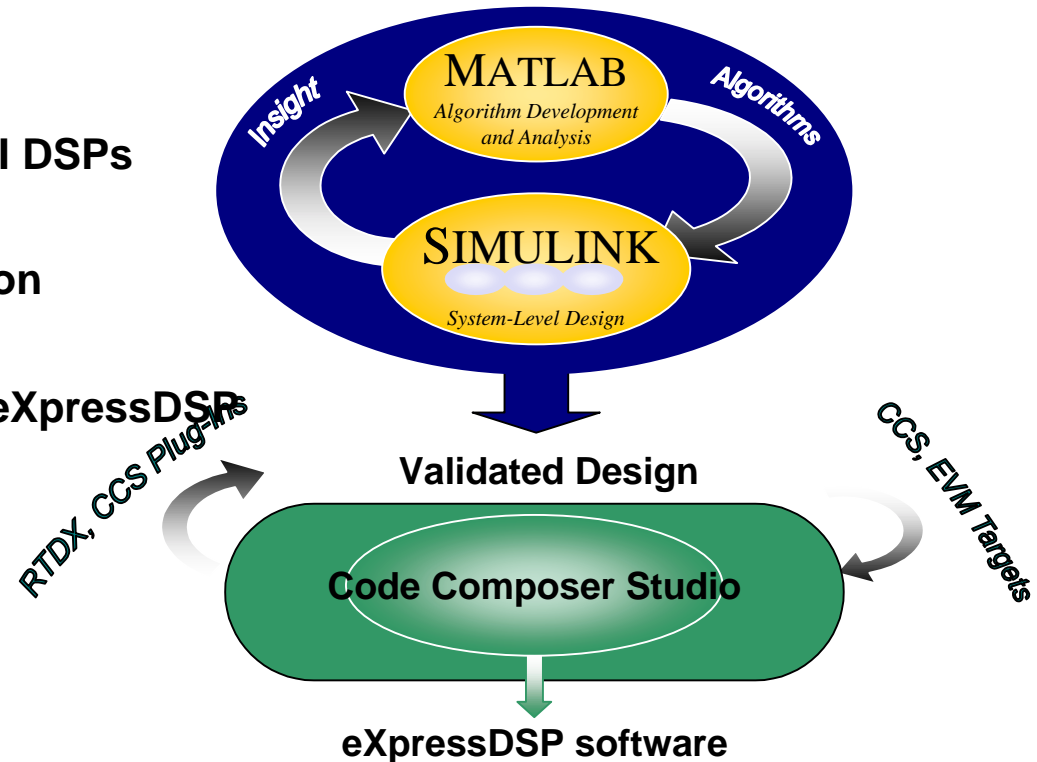


The Next Challenge: Concept-to-Code Integration for eXpressDSP



TI DSP Developer's Kit

- Integrates MATLAB and Simulink with eXpressDSP tools
- Unified top-down design flow for TI DSPs
- Ideal environment for DSP education
- MathWorks is 100% committed to eXpressDSP

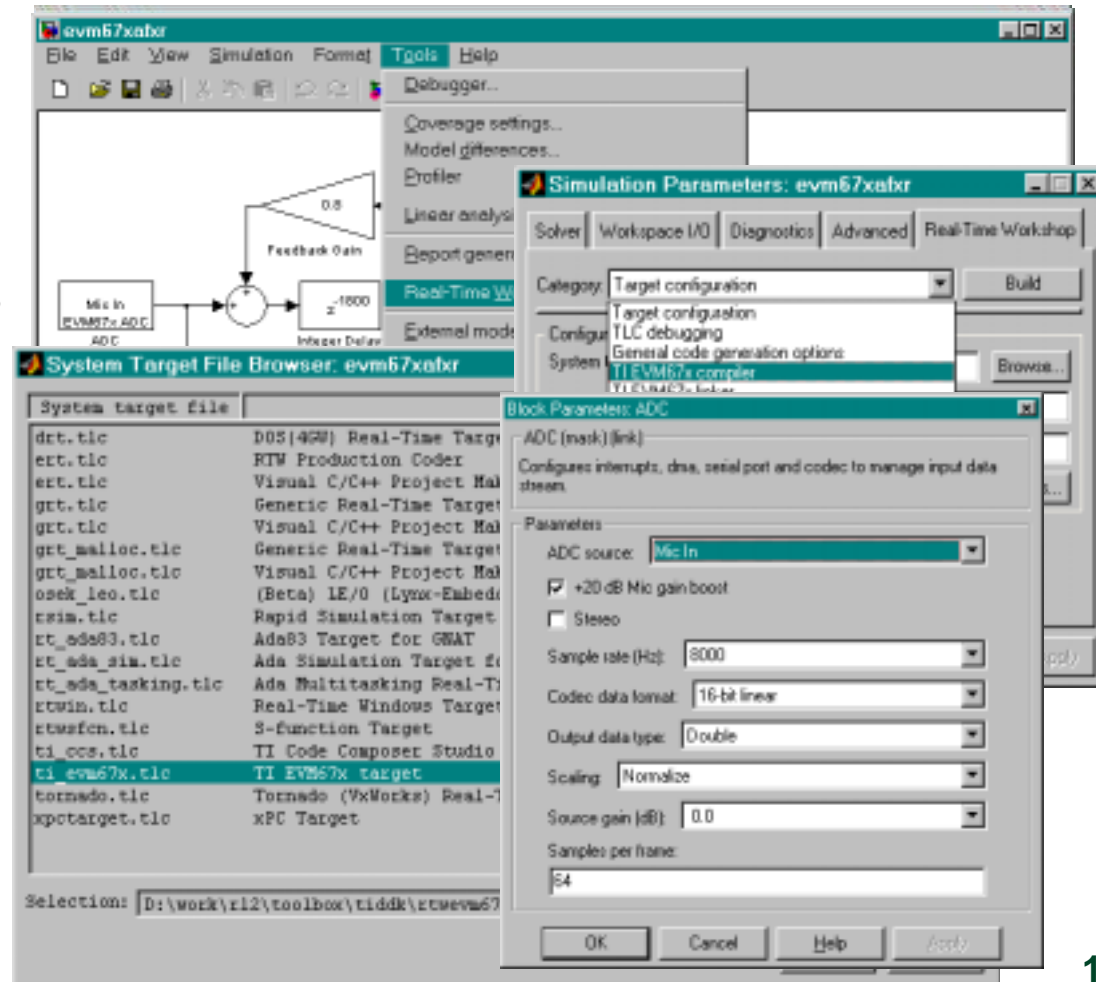


TI DDK Key Features

- **Real-time codegen and rapid prototyping with Simulink**
 - **EVM67x Target**
 - **Code Composer Studio Target**
 - **Benefits**
 - Design graphically in Simulink, deploy and test on TI DSPs
 - Minimize manual algorithm re-coding
- **Real-time analysis and debugging with MATLAB**
 - **Code Composer Studio Automation Interface**
 - **RTDX Interface**
 - **Benefits**
 - Simplify verification and debugging
 - Customize and automate testing, analysis

EVM67x Target

- One-button build & run on C6701 EVM
- Real-time algorithm testing
- Full control of board settings
- Includes codec, LED blocks with source code



Code Composer Studio Target

- Automatically generate C code from Simulink
- Automatically build CCS project
- Integrate with custom code
- Re-target to other development boards

The screenshot shows a Simulink model titled "evm67xabr" with a block diagram for "DSP Blockset Audio Reverb Demo". The diagram includes a "Mic is EVM67x ADC" block, a summing junction, a "Feedback Gain" block with a value of 0.9, a "z^-1000" block labeled "Integer Delay", and an output block. Below the diagram, text says "Type <Ctrl-B> to build and execute model on EVM67 board".

Overlaid on the Simulink window is the "System Target File Browser" for "evm67xabr". It lists various target options:

System target file	Description
drt.tlc	DOS (486) Real-Time Target
ert.tlc	RTW Production Coder
ext.tlc	Visual C/C++ Project Makefile
grr.tlc	Generic Real-Time Target
gvt.tlc	Visual C/C++ Project Makefile
grr_malloc.tlc	Generic Real-Time Target with malloc
gvt_malloc.tlc	Visual C/C++ Project Makefile with malloc
osek_leo.tlc	(Beta) LE/O (Lynx-Embedded)
rsim.tlc	Rapid Simulation Target
rt_ads3.tlc	Ada83 Target for GMAT
rt_ads_sim.tlc	Ada Simulation Target for GMAT
rt_ads_tasking.tlc	Ada Multitasking Real-Time Target
rtwin.tlc	Real-Time Windows Target
rtwfcn.tlc	S-function Target
ti_ccs.tlc	TI Code Composer Studio target
ti_evm67x.tlc	TI EVM67x target
tornado.tlc	Tornado (VxWorks) Real-Time Target
xpcrtarget.tlc	xPC Target

The "ti_ccs.tlc" option is selected, and the path is shown as "D:\work\rl2\toolbox\tiddr\rtwccs\ti_...".

The screenshot shows the Code Composer Studio (CCS) interface. The left pane shows a project tree with files like "evm67xabr.c" and "evm67xabr.h". The right pane shows the generated C code for "evm67xabr.c".

```

// evm67xabr.c
// Real-Time Workshop code generated for Simulink model 'evm67xabr.mdl'.
// Model Version : 1.00
// Real-Time Workshop file version : 4.0 (Date: 2000-06-21 10:00:00)
// Real-Time Workshop file generated on : Wed Jul 11 11:51:42 2001
// TWC version : 3.5 (Apr 10 2001)
// C source code generated on : Wed Jul 11 11:51:42 2001

// Rejected TWC options:
// InlineParameters = 0
// SingleThreaded = 0
// CodeOptim = RealTime

// Simulink model settings:
// Solver = FixedStep
// StartTime = 0.0 s
// StopTime = 1.000 s
// FixedStep = 0.005 s

#include "math.h"
#include "string.h"
#include "evm67xabr.h"
#include "evm67xabr_gsw.h"

// Function: config_serial_port
// Abstract: Remove interrupt callback routine used to regulate the
// timer when in rt_block.

void config_serial_port(int port)
{
    // define control variables unless #
    // user assigned int port
    MAKE_CLEAR_POL_HIGH_CLEARF;
    MAKE_CLEAR_POL_FALLING_EDGEF;
    MAKE_CLEAR_POL_HIGH_FEED;
    MAKE_CLEAR_POL_HIGH_FEEDF;
    MAKE_CLEAR_FEED_EDGE_FEED;
    MAKE_CLEAR_FEED_EDGE_FEEDF;
}
    
```

MATLAB - eXpressDSP Plug-ins

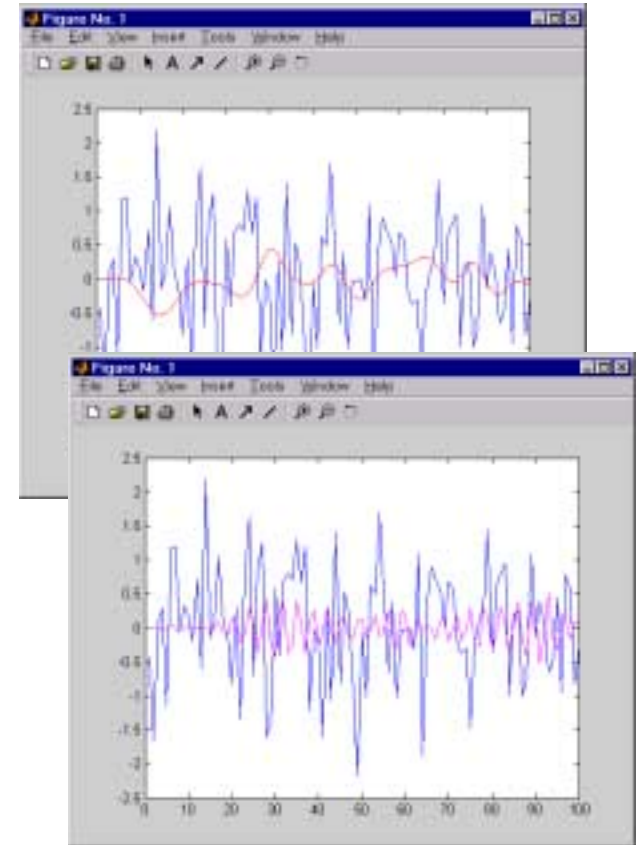
- **Code Composer Studio Plug-in**
 - Access CCS API from MATLAB
 - Compare target outputs to MATLAB or Simulink results
 - Integer and floating point data formats
- **RTDX Plug-in**
 - Programmatic access to RTDX API from MATLAB
 - Real-time analysis and visualization of TI DSP applications
 - Create custom tools and GUIs in MATLAB

Plug-in Example

```
% Activate Link to Code Composer
target=ccsdsp;
target.cccd( 'D:\Work\RTDX\Target\DemoFIR' );
target.ccdir;
% Load Workspace and Target Code
target.open( 'DemoFIR.wks' );
target.open( 'DemoFIR.out' );

[...]
% Use Matlab to design a low-pass filter and
transfer to DSP
new_coeff = fir1(20,0.2);
target.write(addr_coeff{2},new_coeff);

% Run the filter and read the results
target.run( 'runtohalt',20.0);
filt_dout =
target.read(addr_dout{2},size(test_din),'double');
plot(1:100,test_din,'b-',1:100,filt_dout,'r-');
```



Summary

- **Simulink system-level DSP design solutions**
- **New eXpressDSP plug-in shipping in Q4**
- **Top-down design flow for TI DSPs**
- **Integrates tools from industry leaders**
- **Bridges the gap between algorithm development and implementation**